Strxfrm, Wcsxfrm

Carefully manage buffer sizing and units. Ensure entire string is transformed.

Sean Barnum, Cigital, Inc. [vita[1]]

2007-04-23

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 6523 bytes

Attack Category	- Malicious Input - Denial of Service - Privilege Exploitation
Vulnerability Category	- Buffer Overflow - No Null Termination
Software Context	- String Management - String Conversion MACROS
Location	- string.h
Description	When using the string transform functions strxfrm() or wcsxfrm(), problems can result if care is not taken to ensure that the entire input string is transformed into a correctly sized buffer. The strxfrm() function transforms a string so that strcmp() can be used for lexical comparisons, taking into consideration the value of LC_COLLATE. The transformations performed by strxfrm() are such that, if two strings are transformed, the lexical relationship of the transformed strings as determined by strcmp() is the same as the lexical relationship of the original strings as determined by strcoll(). wcsxfrm() is a wide-character version of strxfrm(); the string arguments of wcsxfrm are wide-character pointers. For wcsxfrm, after the string transformation, a call to wcscmp with the two transformed strings yields results identical to those of a call to wcscoll applied to the original two strings. wcsxfrm and strxfrm behave identically otherwise. These functions are subject to buffer overflow if the buffer size is not correctly specified. Particular care must be taken to ensure correct sizing when wide characters are used. The results are indeterminate and the output string may not be null terminated if the entire string and terminating null could not be transformed.

[1] http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

<table>
<tr><td></td><td>Therefore, it is important to ensure that the entire string is transformed.</td></tr>
<tr><td><b>APIs</b></td><td><table><tr><th>Function Name</th><th>Comments</th></tr><tr><td>_tcsxfrm</td><td>Generic function (Windows)</td></tr><tr><td>strxfrm</td><td>ASCII implementation</td></tr><tr><td>wcsxsfrm</td><td>Unicode implementation</td></tr></table></td></tr>
<tr><td><b>Method of Attack</b></td><td>If "count" parameter overestimates the buffer size, an attacker who controls the input string to be transformed can arrange for a buffer overflow and potential achieve arbitrary code execution.
Even if the "count" parameter corresponds to the buffer size, failure to guarantee null termination of the result can lead to unexpected behavior from a subsequent call to strcmp(). An illegal memory access might result in program termination, or other unexpected behavior could result. These conditions may result in a denial of service or expose some other vulnerability that an attacker could exploit.</td></tr>
<tr><td><b>Exception Criteria</b></td><td></td></tr>
<tr><td><b>Solutions</b></td><td><table><tr><th>Solution Applicability</th><th>Solution Description</th><th>Solution Efficacy</th></tr><tr><td>Whenever the indicated functions are called.</td><td>Ensure that the entire input string and null termination are converted. This can be done by checking the return value and enlarging the buffer if the tranform was incomplete, or by doing a dummy conversion first to determined the needed buffer size, sizing the buffer accordingly, then performing the transform.
Ensure that the specified maximum character count</td><td>Effective.</td></tr></table></td></tr>
</table>

	reflects the buffer size. Remember that for wide characters, the size of the buffer in characters is not equal to the size in bytes.

Signature Details	size_t strxfrm(char *strDest, const char *strSource, size_t count);
	size_t wcsxfrm(wchar_t *strDest, const wchar_t *strSource, size_t count);

Examples of Incorrect Code	char strSource[] = "Some text to be transformed for collating.";
	char strDest[20];
	strxfrm(strDest, strSource, 21); // Count exceeds buffer size - buffer will overflow
	// The following is likely to go awry because complete string was not transformed,
	// and result may not be null terminated.
	if (strcmp(strDest, comparisonString) > 0) { /* act based on comparison */ }

Examples of Corrected Code	char strSource[] = "Some text to be transformed for collating.";
	// allocate a buffer as large as it needs to be to contain result
	int charsToProduce = strxfrm(NULL, strSource, 0)+1;
	if (charsToProduce == 0) { /* handle error */ }
	char *strDest = (char *)malloc(charsToProduce *sizeof(char));
	strxfrm(strDest, strSource, charsToProduce);
	if (strcmp(strDest, comparisonString) > 0) { /* act based on comparison */ }

Source Reference	• Rough Auditing Tool for Security (RATS)[2]
Recommended Resources	• Man page for strxfrm()[3] • Man page for wcsxfrm[4]

<table>
<tr><td></td><td>• MSDN reference for strxfrm, wcsxfrm, _tcsxfrm[5]</td></tr>
<tr><td><b>Discriminant Set</b></td><td><table><tr><td><b>Operating System</b></td><td>• Any</td></tr><tr><td><b>Languages</b></td><td>• C
• C++</td></tr></table></td></tr>
</table>

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

[1] mailto:copyright@cigital.com